In the claims:

1. (currently amended) A method for operating a second, sending network node relative to ~~determining a steady-state operating point of~~ a first, receiving network node, ~~relative to a second network node~~ the receiving node being operable to receive data packets sent by the sending node via a plurality of flows, the method comprising:

determining a queue law function of the first, receiving node based upon predetermined system traffic conditions, the queue law function being representative of average queue size of an exemplary queue of the receiving node based on traffic conditions and percentage of dropped packets; and

determining a control function for the second, sending node based upon the queue law function, the control function defining a drop percentage when a buffer of the second node is filled above a predetermined threshold based upon an average queue size, wherein the control function prompts a gradual increase of drop probability of all data packets in flows bound for the receiving network node in an overload condition defined by intersection of the queue law function and the control function,

whereby each flow between the sending node and receiving node is proportionally affected regardless of sending rate of particular flows.

2. (original) A method according to claim 1, wherein the control function is a random early detection control function.

3. (original) A method according to claim 1, wherein the control function does not have a non-bounded discontinuity.

4. (original) A method according to claim 1, wherein the control function comprises two piecewise linear segments.

5. (currently amended) A method for modeling dynamics of a queue in a first, sending node having a buffer in order to determine a steady-state operating point of the first node relative to a

second, receiving node, the receiving node being operable to receive data packets sent by the sending node via a plurality of flows, the method comprising:

calculating a queue law function dependent on traffic conditions at the second node, the queue law function being representative of average queue size of an exemplary queue of the receiving node based on traffic conditions and percentage of dropped packets; and

determining a point of operation for the first node as the intersection of the queue law function and a predetermined control function for the first node, the control function defining a drop percentage when a buffer of the second node is filled above a predetermined threshold based upon an average queue size,

wherein the control function increases drop probability of all data packets in flows bound for the receiving network node gradually in an overload condition,

whereby each flow between the sending node and receiving node is proportionally affected regardless of sending rate of particular flows.

6. (original) A method according to claim 5, wherein the point of operation defines a packet drop percentage for dropping a percentage of packets from the buffer.

7. (original) A method for improving congestion control according to claim 5, wherein the node resides in a network.

8. (original) A method for improving congestion control according to claim 7, wherein the network operates in a TCP environment.

9. (original) A method for improving congestion control according to claim 5, wherein data received at a node is acknowledged.

10. (original) A method according to claim 5, wherein the traffic conditions which determine the queue law function are the number of flows into the node, an average packet size and an average round trip transmission time.

11. (original) A method according to claim 5, wherein the function is determined by assuming that the node does not experience feedback.

12. (original) A method according to claim 5, wherein the point of operation determines a drop rate.

13. (original) A method according to claim 5, further comprising:

dropping packets from the buffer at the determined drop rate.

14. (cancelled)

15. (cancelled)

16. (cancelled)

17. (cancelled)

18. (cancelled)

19. (cancelled)

20. (cancelled)

21. (cancelled)

22. (cancelled)

23. (cancelled)

24. (cancelled)

25. (cancelled)

26. (cancelled)

27. (cancelled)

28. (cancelled)

29. (cancelled)

30. (cancelled)

31. (cancelled)

32. (cancelled)

33. (cancelled)

34. (cancelled)

35. (cancelled)

36. (cancelled)

37. (cancelled)

38. (cancelled)

39. (cancelled)

40. (cancelled)

41. (cancelled)

42. (cancelled)

43. (cancelled)

44. (cancelled)

45. (cancelled)

46. (cancelled)

47. (cancelled)

48. (cancelled)

49. (cancelled)

50. (cancelled)

51. (cancelled)

52. (cancelled)

53. (cancelled)

54. (cancelled)

55. (cancelled)

56. (cancelled)

57. (cancelled)

58. (cancelled)

59. (cancelled)

60. (cancelled)

61. (cancelled)

62. (cancelled)

63. (cancelled)

64. (cancelled)

65. (cancelled)

66. (cancelled)

67. (cancelled)

68. (cancelled)

69. (cancelled)

70. (cancelled)

71. (cancelled)

72. (cancelled)

73. (cancelled)

74. (cancelled)

75. (cancelled)

76. (cancelled)

77. (cancelled)

78. (cancelled)

79. (cancelled)

80. (cancelled)

81. (cancelled)

82. (cancelled)

83. (cancelled)

84. (cancelled)

85. (cancelled)

86. (cancelled)

87. (cancelled)

88. (cancelled)

89. (cancelled)

90. (cancelled)

91. (cancelled)

92. (cancelled)